Real-time Control Software for Optical Interferometers: The RICST Testbed

Richard L. Johnson Jr.[1], Elizabeth A. McKenney[2] Kenneth M. Starr[3]

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91009

ABSTRACT

RICST (pronounced "rixt"), the Realtime Interferometer Control System Testbed, is a testbed at NASA's Jet Propulsion Laboratory where teams are developing a common electronics and software foundation that will form the core of the control system for JPL's various interferometer projects.

Important technologies being developed and demonstrated on the RICST testbed include hard real-time multiprocessing, generic interferometer software and hardware architecture, easy reconfigurability, abstract constructs for easy implementation of control loops and servos, and highly configurable telemetry streams. The basic hardware and software architecture developed by RICST can be applied to virtually any interferometer based on the combiner-collector architecture.

The RICST project began in May 1996 and has been delivering control system "increments" every few months since that time. Each increment adds new end-to-end functionality. White-light fringe tracking has been demonstrated in the lab already; the delivery of a fully operational testbed is scheduled for the end of April. Future enhancements will add support for a wide array of calibration, sequencing, and other support functions. RICST deliveries include support software, such as a graphical user interface and a prototype spacecraft bus interface, as well as the embedded software needed to run the instrument.

The RICST program will make final deliveries to its various customers in 2000, after which the RICST testbed facility will continue on in support of the SIM mission.

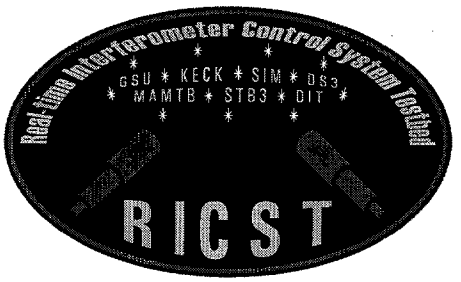
Keywords: interferometer control, real-time software, gizmo, inter-processor communication, servo, controller, telemetry

1. RICST CUSTOMERS

RICST software and electronics are designed to be a highly modular and flexible package, which can be easily modified and reused in a variety of Interferometer projects. JPL's Interferometer Technology Program (ITP) Plans to use RICST in its Micro-Arcsecond Metrology Testbed (MAMTB) and SIM Testbed (STB3) which will lead up to the Space Interferometer Mission (SIM) for which RICST software and electronics will also be used.

The Keck Interferometer will also use RICST as the core for its software and electronics. Georgia State University (GSU) will be using parts of the RICST Delay Line software and electronics in its Interferometer on Mt. Wilson.

RICST is also being planned as the core for a New Millennium Deep Space 3 mission (DS3), a formation flying separated Interferometer.

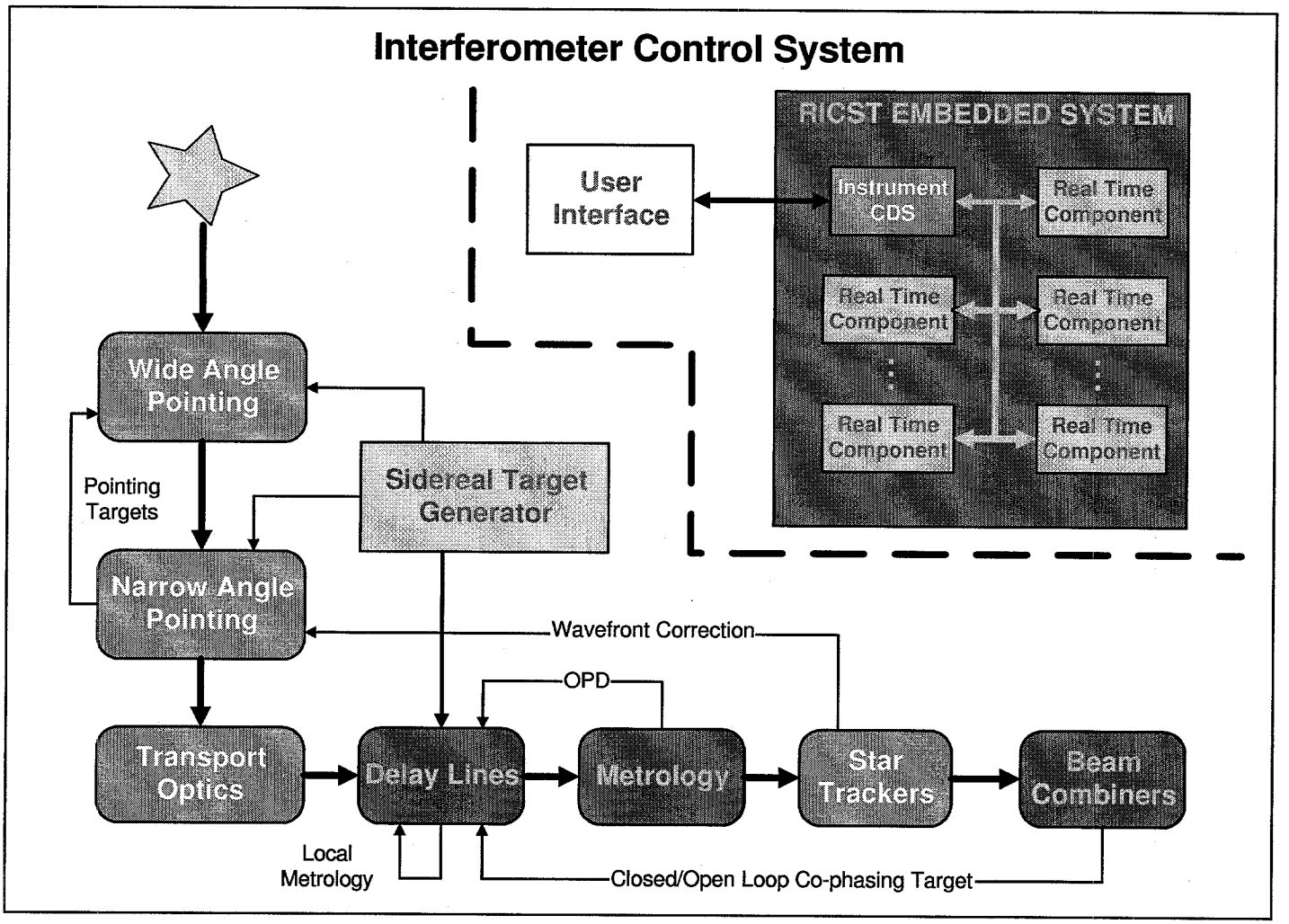


[1] R.L.J: e-mail: rich@huey.jpl.nasa.gov ; Telephone: (818) 354-1772
[2] E.A.M: e-mail: mckenn@huey.jpl.nasa.gov ; Telephone: (818) 354-5136
[3] K.M.S: e-mail: ken@huey.jpl.nasa.gov ; Telephone: (818) 393-2980

2. RICST GENERIC ARCHITECTURE

Though each RICST interferometer customer has its own set of functional requirements there exists a broad set of properties in common that makes it possible to develop common software. All are based on a "combiner-collector" architecture, wherein remote collectors relay light to a central station for combination. All have collectors, fast tip-tilt control systems, delay lines, fringe trackers, and sequencing and telemetry capability. All require tight synchronization of hardware and software events.

At its core, RICST identifies and defines key interfaces and components common to all interferometers thus providing a set of building blocks to construct control systems for each specialized application. Using object oriented software design concepts and techniques RICST encapsulates these components into generic objects that specify basic interfaces and behaviors. By deriving specialized forms of these generic objects, RICST or a RICST customer is able to tailor the control system to meet unique requirements without having to re-invent the entire component. This inherent re-use of common functionality reduces the number of lines of code needed to implement a new real-time control component – and fewer lines of code to write means fewer bugs and easier maintenance.

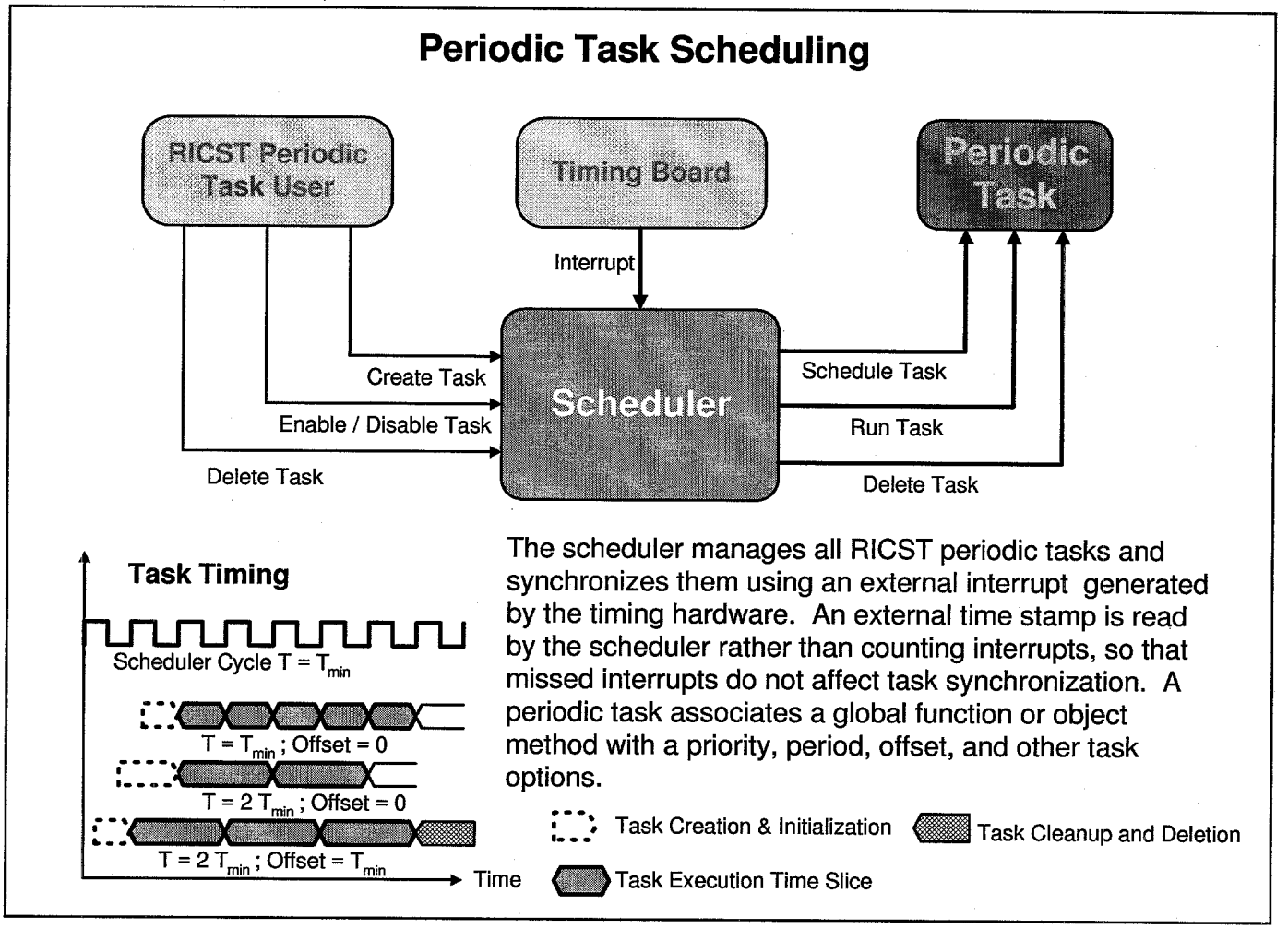


3. PERIODIC TASK SCHEDULING

The interferometer control system has a number of requirements that translate into the need to execute the same piece of code repeatedly, at a precise rate. A typical example is code that implements a digital servo in the computer. At a fixed rate (e.g., perhaps 100 Hz for a motor control servo), the computer must read a sensor, perform a control law calculation, and then output to an actuator.

The Scheduler is RICST's approach to this requirement. The Scheduler is a system that can be configured at run-time to repeatedly call one or more functions, or tasks, at rates that are factors of some base rate at which the whole Scheduler runs. For example, the Scheduler might run at a base rate of 1000 Hz, while calling the function DoFastServo at 1000 Hz, DoSlowServo at 200 Hz, and CheckLimitSwitches at 10 Hz. A user of the Scheduler sets up which functions will be called and at what frequencies.

The Scheduler handles the scheduling of tasks by responding to clock interrupts. At each interrupt, it observes the present time and determines whether any of the tasks it is managing should be scheduled to run, marking each such task as "ready". It then steps through the list of "ready" tasks (ordered by priority) and starts each task running. The operating system then handles the sharing of CPU time among the running tasks.



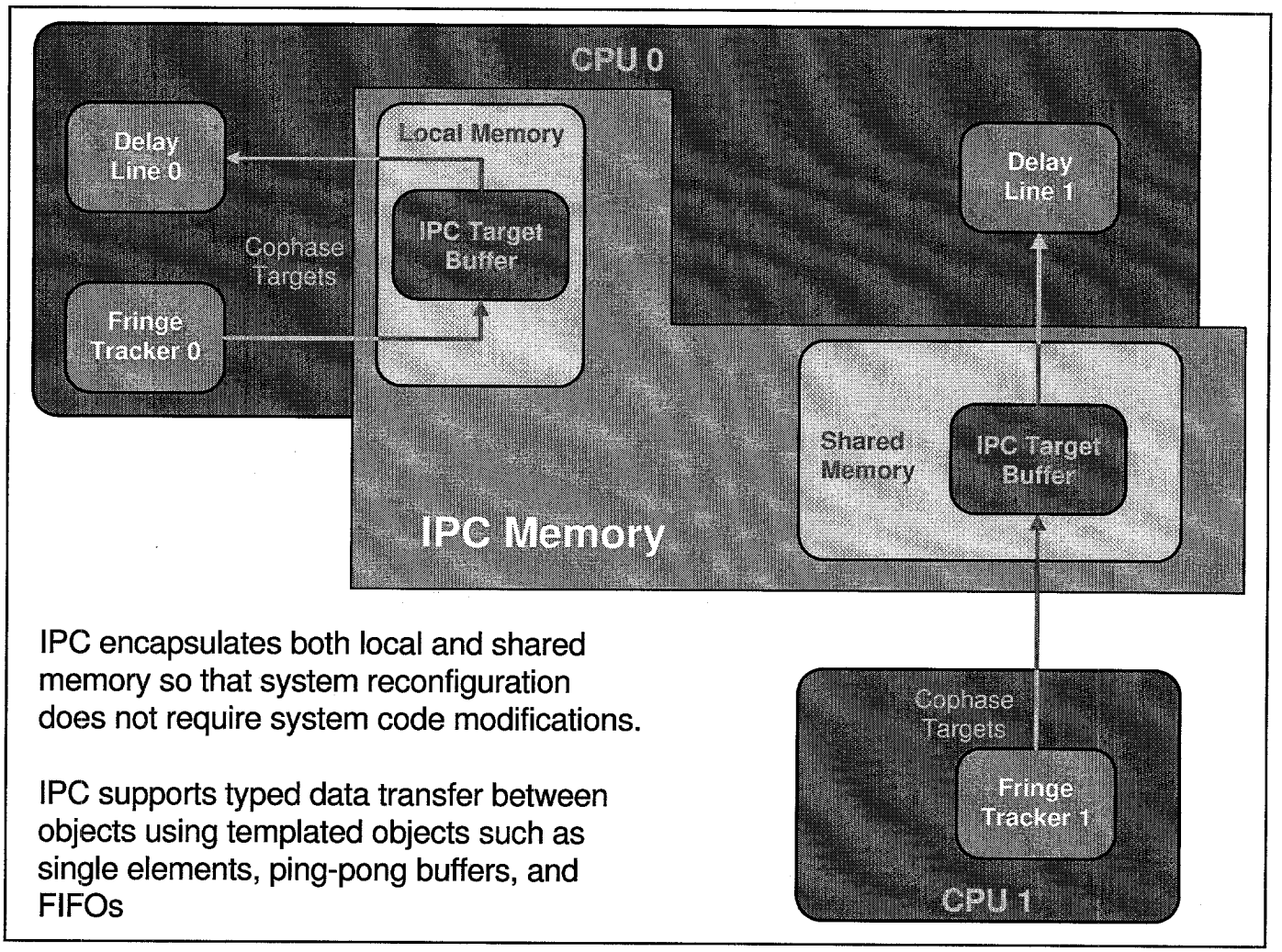
4. INTER-PROCESSOR COMMUNICATIONS

Inter-processor Communication (IPC) provides a mechanism for allocating and accessing data objects that are shared between RICST components. Because the sharing components may reside on the same or on different processors, IPC encapsulates both local and shared memory using the identical interface thus allowing easier reconfiguration of sharing components.

IPC addresses the problem of maintaining a persistent mapping of shared data locations should one or more CPUs or process needs resetting by requiring that all users of an IPC object allocate that object. IPC maintains a header list of all IPC objects and only physically allocates objects that do not yet exist. The allocator in either case receives the memory location of the shared data.

In order to prevent memory corruption due to simultaneous allocation by different processes, all local memory allocations are protected using semaphore blocking. In the case of shared memory allocation, a custom round-robin arbiter grants different CPUs permission to allocate an IPC object.

IPC utilizes C++ template classes so that almost any data object (except those with pointers or v-tables) can be an IPC object. IPC currently provides six types of shared data encapsulators: IPC Elements contain a single instance of the shared data and does not allow handle simultaneous writes or read/write operations. IPC Double Buffers and Indirection Double Buffers implement a ping-pong buffer that handles simultaneous read/write operations. The indirection double buffer provide transport for byte arrays. IPC FIFOs and Multi-type FIFOs implement a queue that allows simultaneous read/write operations. The multi-type FIFO provide a pseudo-typed transport mechanism for data of different types.



5. SERVOS AND CONTROLLERS

Real-time control software relies on both open loop and closed loop sensing and actuation of the physical devices or "plants." The RICST servo architecture provides a framework of periodic tasks and controller objects that allow implementation of specialized control laws that have similar interfaces.

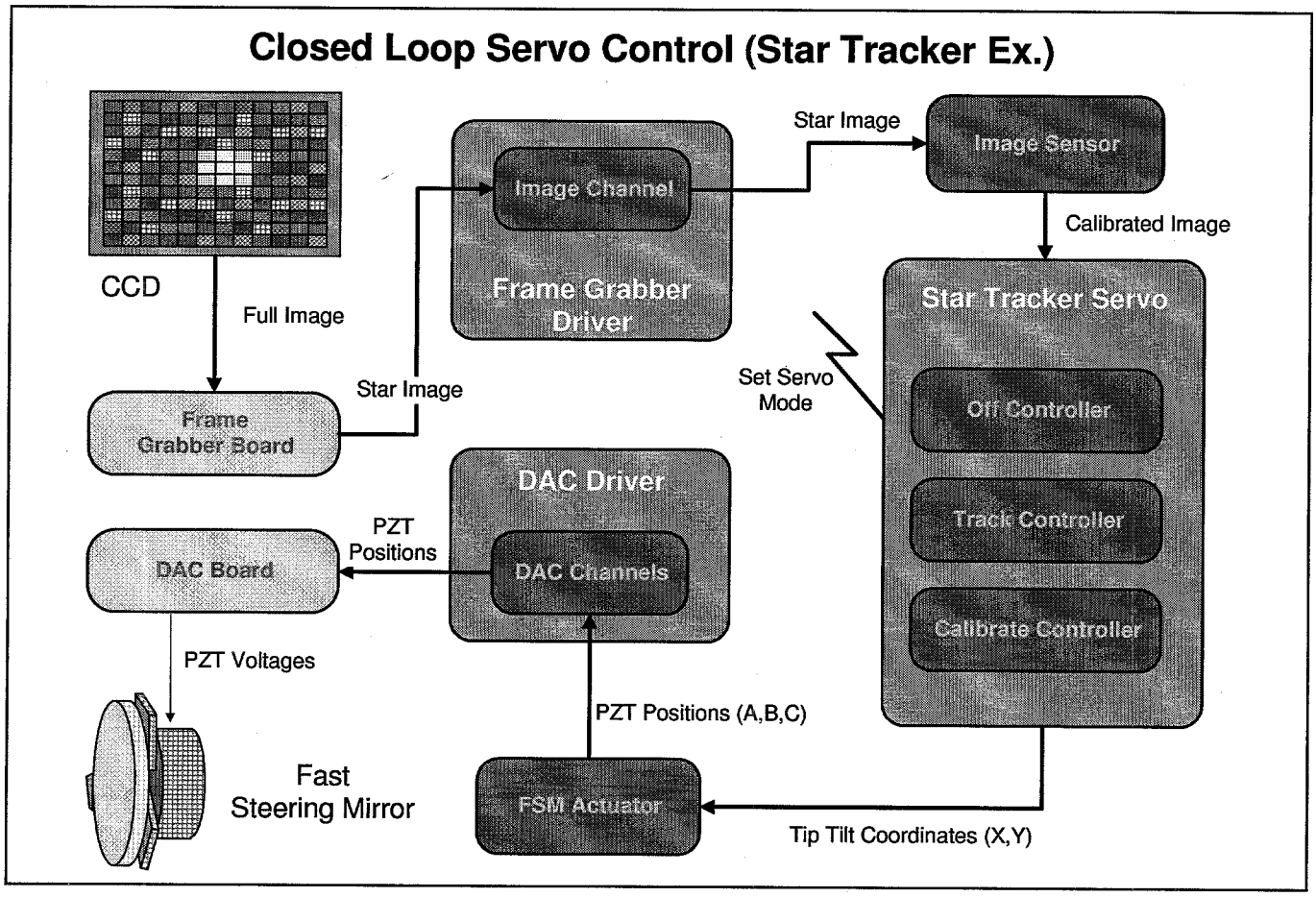
RICST servos may be simple – having only a single periodic task and a few controllers, or complex – consisting of multiple interconnected periodic tasks and controller with behaviors that vary depending on internal and external states.

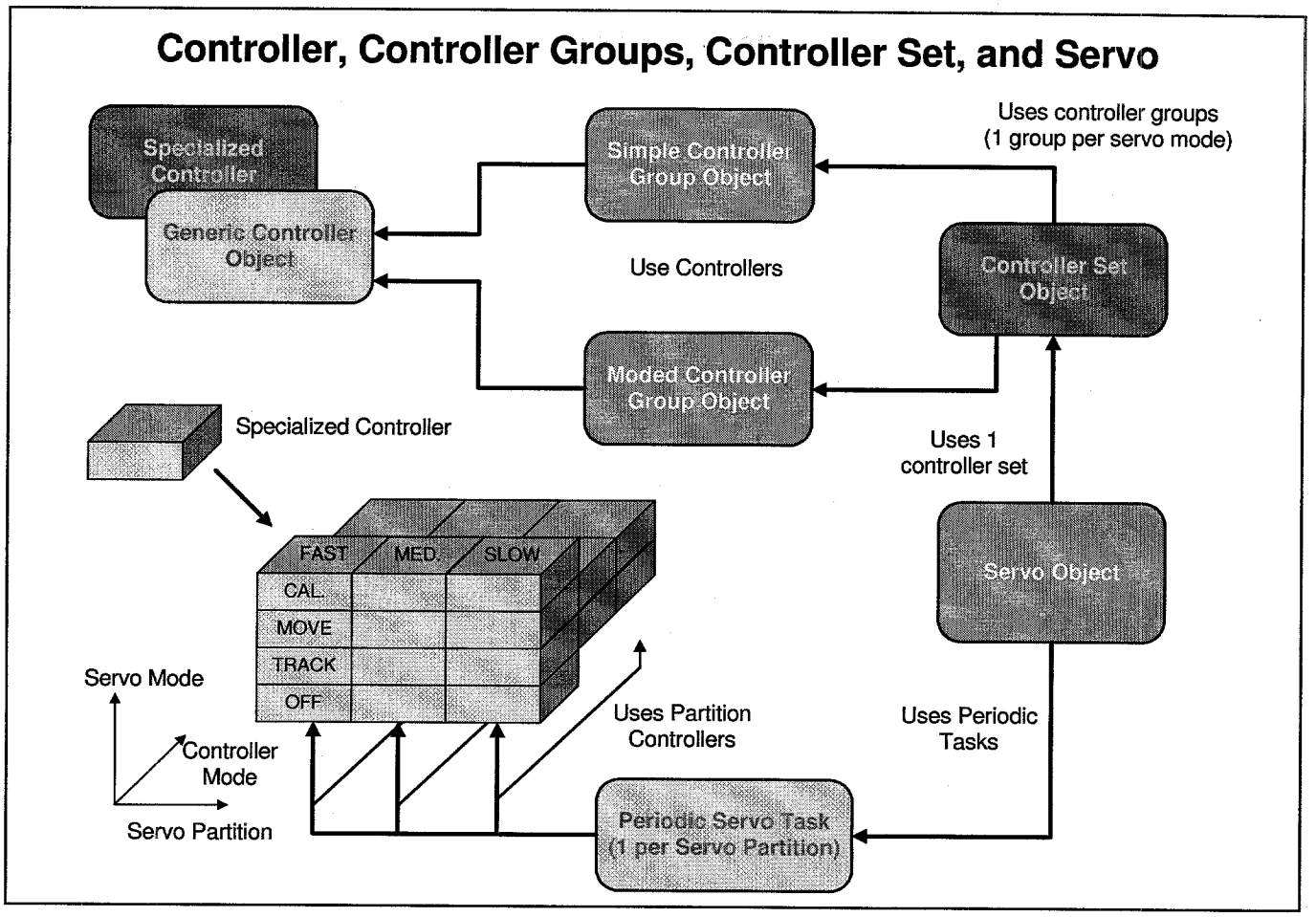
A RICST servo is an organized container of control laws. In general, there are three dimensions to the servo: Servo Mode, Servo Partition, and Controller Mode.

A servo mode is a user selectable servo state that corresponds to functionally different servo behavior. For example the delay line servo has a Front servo mode that moves the delay line to the front of the track, and a Track servo mode that performs nano-meter OPD control to desired target position.

Servo partitions correspond to individual periodic tasks that run simultaneously at various rates. Each servo mode has a controller for each partition so that a change in servo mode corresponds to a controller change in all partitions. This set of controllers forms a controller group for each servo mode. In the case of the delay line, three servo partitions are coupled to control the total OPD. The fastest partition (5 kHz) controls a PZT actuator using a laser metrology sensor for the nano-meter OPD correction. A medium rate partition (1 kHz) controls one or more voice coil actuators to keep the PZT within it's dynamic range. Similarly the slowest partition (100 Hz) controls a motor to keep the voice coils within their dynamic range.

Controller Modes add additionally control flexibility for particular servo modes. A controller mode corresponds to a particular type of control used for a given servo mode and partition under specified internal state conditions. For example, the delay line track servo mode has two controller modes: Track and Slew. The track controller mode performs nano-meter OPD closed loop tracking using PZT, voice coils, and motor. The slew controller mode moves the delay line cart close to the desired target as fast as possible using only the motor. The active controller mode depends on the measured OPD error – Slew if the delay line is far from the target, Track if near the target. Like servo modes, changes to the controller mode affect all servo partitions. Unlike servo mode changes that are typically event driven, controller mode changes are instigated by a periodic task that monitors appropriate states. This periodic task is owned and managed by the specific controller group. In general a servo may contain both controller groups with (moded controller groups) and without (simple controller groups) controller modes.



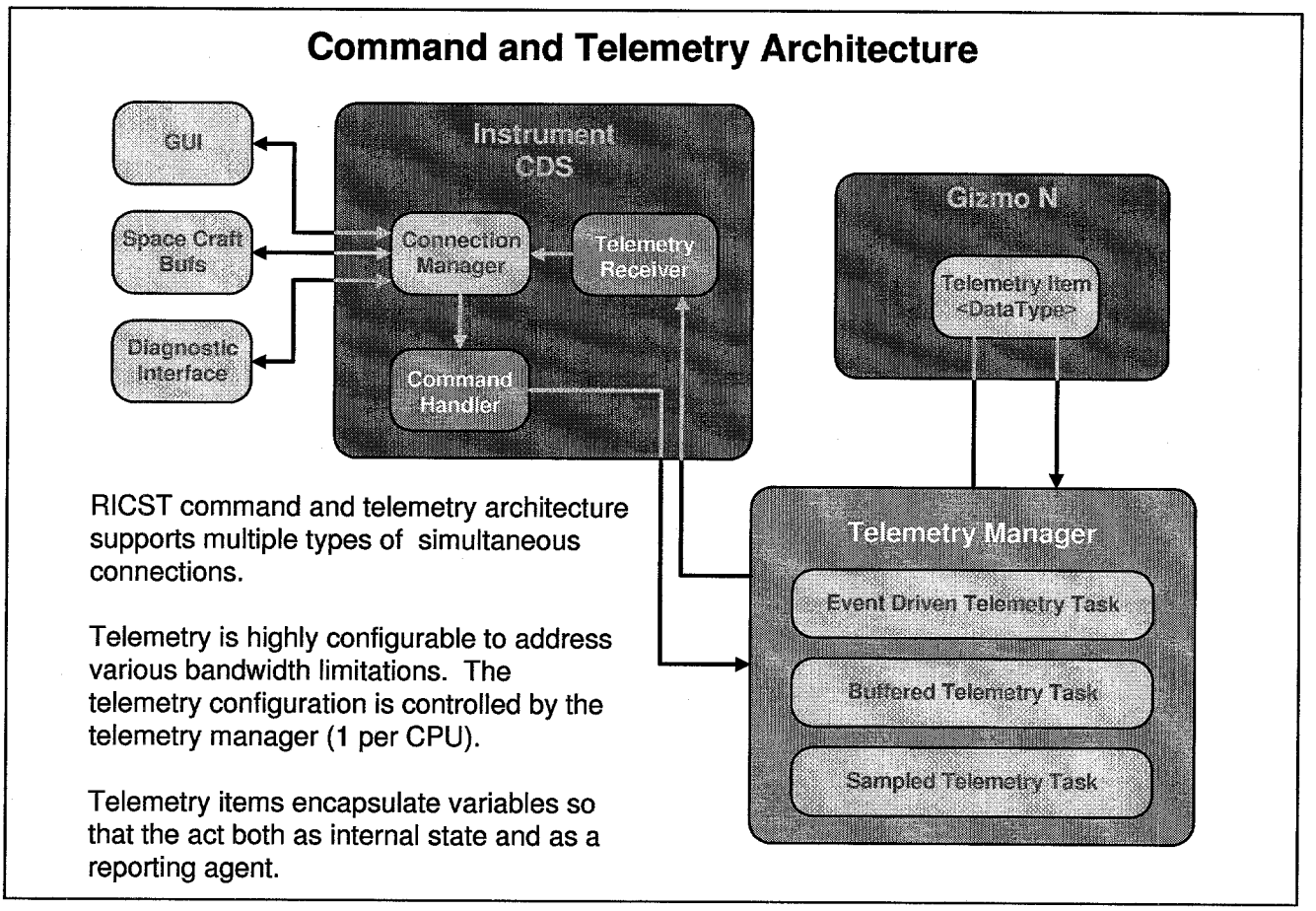

Controller, Controller Groups, Controller Set, and Servo
Specialized Controller
Generic Controller Object
Simple Controller Group Object
Uses controller groups (1 group per servo mode)
Use Controllers
Controller Set Object
Moded Controller Group Object
Specialized Controller
Uses 1 controller set
FAST
MED.
SLOW
CAL
MOVE
TRACK
OFF
Servo Object
Servo Mode
Controller Mode
Servo Partition
Uses Partition Controllers
Uses Periodic Tasks
Periodic Servo Task (1 per Servo Partition)

6. CONFIGURABLE TELEMETRY

Interferometers present a big challenge for traditional telemetry systems (especially space based) because of the enormous quantity of potential scientific data and engineering diagnostics. Because the complete set of all telemetry exceeds bandwidth limitations for all current communication links and because different users desire different telemetry subsets, RICST will provide a highly configurable telemetry architecture.

RICST identifies three fundamental types of telemetry: **Event Driven** telemetry is information that is reported whenever a particular event occurs. This event may be a value change or other complex conditions. **Buffered** telemetry is typically data that is generated at a known rate (do to a periodic task) and for efficiency reasons is buffered rather than sending each data element individually. **Sampled** telemetry is data that is sent to the user at a specified rate. This type of data is most likely used as status information on a GUI.

The telemetry design components include: a **Connection Manager** to receive commands and delivers telemetry to one or more user interface, a **Telemetry Manager** that configures and processes telemetry on each CPU, a **Telemetry Receiver** that multiplexes telemetry from each manager, and **Telemetry Items** that encapsulate data or diagnostics to be sent to the user.



7. RICST GIZMOS

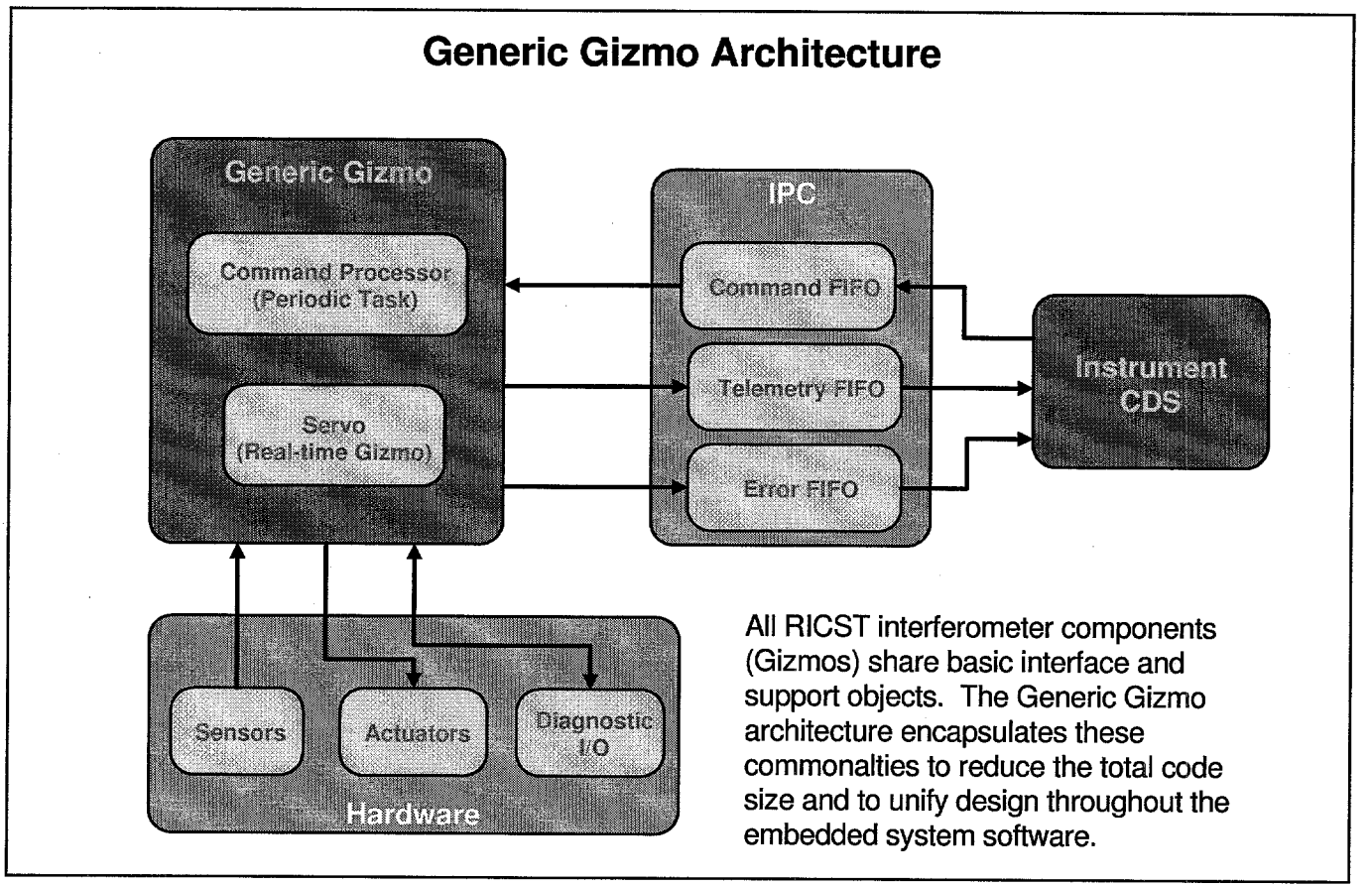
A RICST Gizmo encapsulates a functional interferometer component such as a delay line, fringe tracker, star tracker etc. If a given interferometer has six physical delay lines the embedded system software would instantiate six delay line gizmos. In general a gizmo may be abstracted as an embedded system component that:

1) Receives commands from the Instrument Command Data System (ICDS -another specialized gizmo)
2) Reports telemetry and errors to the ICDS.

3) Receives inputs from hardware sensors and diagnostic channels.
4) Actuates hardware actuators and diagnostic channels.
5) May have a servo with associated controllers.
6) May communicate with other gizmos using IPC objects.

These similarities among gizmos enables RICST software to define two base class gizmos to hide implementation of common functionality: tGizmo and tRealtimeGizmo.

The tGizmo class is the base class for all RICST gizmos including tRealtimeGizmo. tGizmo encapsulates command processing by controlling access to the command FIFO. Derived gizmos need only to overload a single method to define appropriate responses to commands particular to that gizmo. The tRealtimeGizmo class is the base class for all gizmos that require a servo. Derived gizmos need only specify the servo dimensions and the arrangement of controllers within the servo.

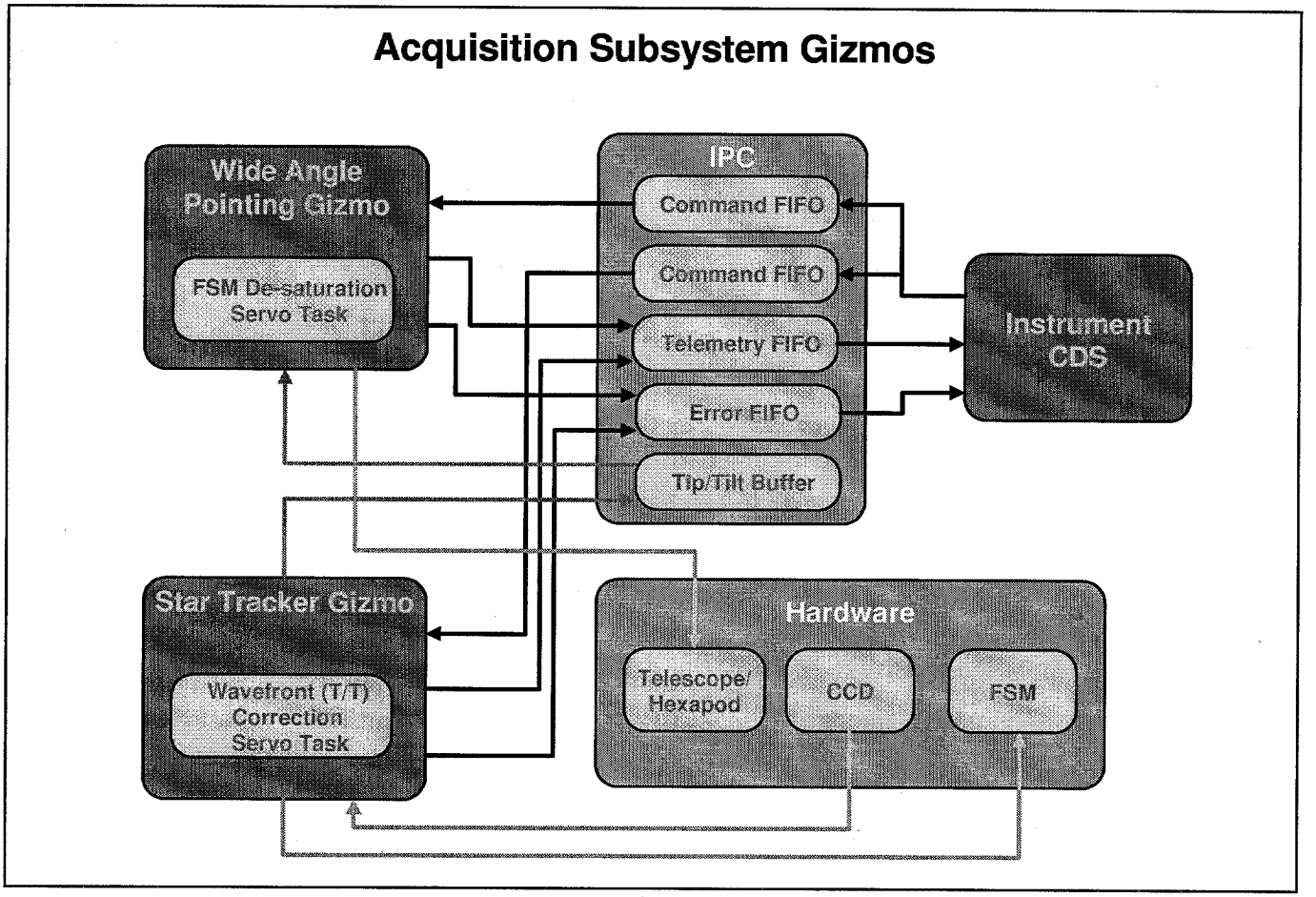


8. ACQUISITION GIZMOS

For each baseline, an interferometer combines the light from two images of the same star to form an interference pattern. Each of the images is derived from a separate optical path, each of which contains a collector (with high dynamic range) and a fast steering mirror (FSM; of limited range but fast response). The two star images are formed separately on one CCD, and are combined for interference on another detector. Each FSM is controlled by a Star Tracker (ST) Gizmo, and each collector is controlled by a Wide Angle Pointing Gizmo (WAP). The purpose of the Star tracker is to keep the two interfering wave fronts parallel to within a fraction of an arc-second. The purpose of the Wide Angle Pointing Device is to keep the corresponding steering mirrors centered to avoid saturation and to minimize any non-linear effects across the fast steering mirror surface.

Each Star Tracker Gizmo monitors the position of a star in the input sub-image and generates pointing commands to the FSM, calculated to keep the star centroid stationary on the detector.

Each Wide Angle Pointing Gizmo monitors the position of a FSM from the Star Tracker and generates pointing commands to the collector mirror, calculated to keep the FSM on its home.

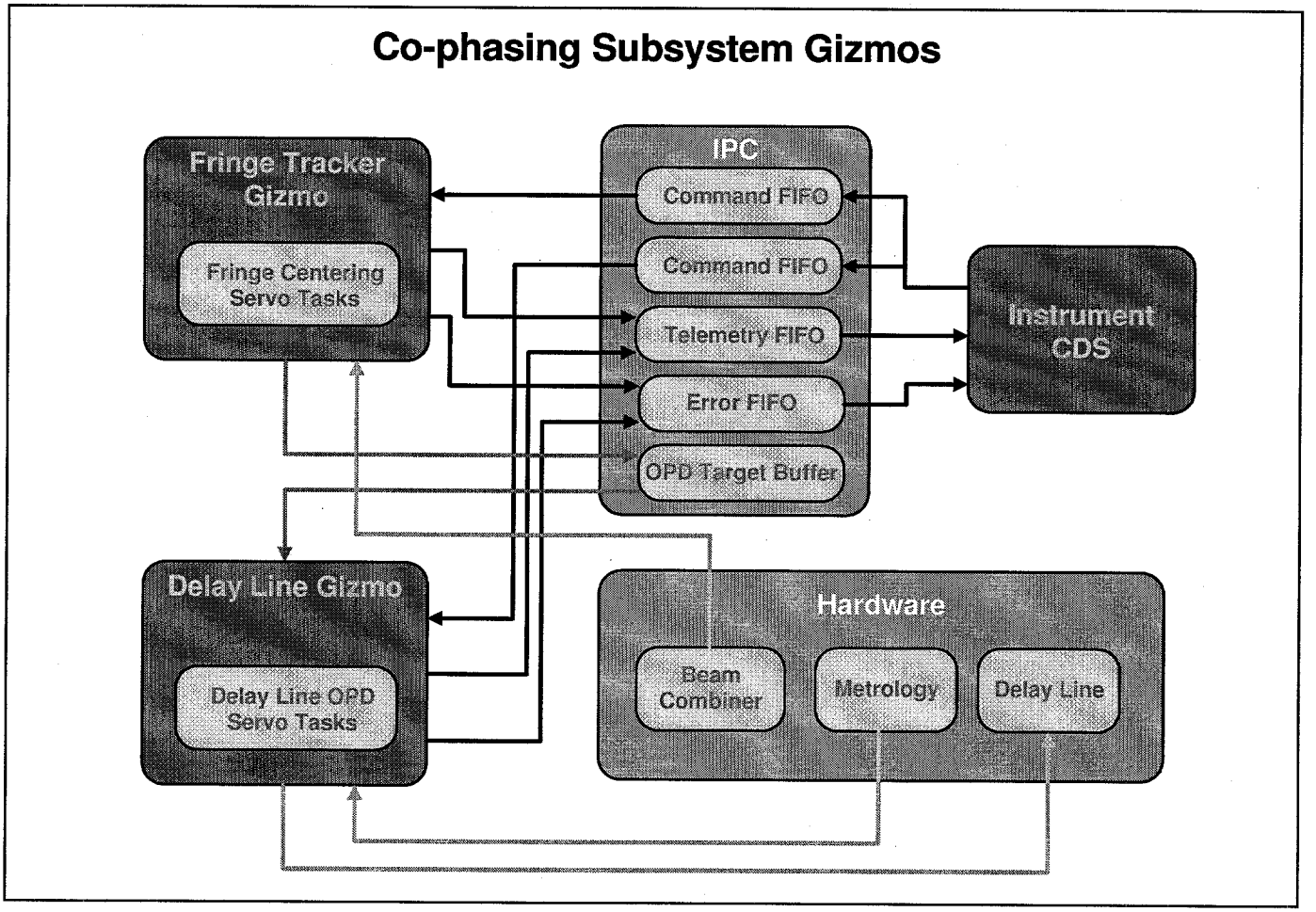


9. CO-PHASING GIZMOS

One of RICST's major requirements is to deliver the infrastructure for nano-meter white-light fringe tracking control software. The two RICST components that address this control issue are the Delay Line Gizmo and the Fringe Tracker Gizmo.

The delay line gizmo encapsulates optical path delay by translating delay line targets (position, velocity, time) into a combination of PZT, voice coil, and motor actuation to achieve the desired OPD. The delay line can accept multiple target sources and tracks the sum of all active targets. The delay line servo consists of three partitions Fast (5 kHz), Medium (1 kHz), Slow (100 Hz) that close the PZT, voice coil (s), and motor loops respectively using internal laser metrology to provide OPD feedback to the controllers.

The fringe tracker gizmo encapsulates the beam combiner hardware and measures fringe position using either photon counting or group delay. The fringe tracker computes delay line targets to center the delay line on the center white-light fringe. In general, a fringe tracker may send targets to multiple delay lines with the targets weighted by a transformation matrix. Such will be the case with the Keck Interferometer in where each of the five fringe trackers will send targets to four delay lines (2 primary star closed loop, 2 secondary star open loop).



10. ACKNOWLEDGEMENTS

This work was carried out by the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronotics and Space Administration and supported by the Interferometer Technology Program – Real-Time Control (ITP-RTC).